LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
Il6r
no. 565-570

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAR 2 2 1978

DEC 1 4 1976

DEC 1 2 REC'D

L161—O-1096

0.84
UIUCDCS-R-73-567
Math
6r
567
0.2

CALCULATION OF GFSR PSEUDORANDOM NUMBER BINARY STARTING MATRIX

by

W. H. Payne

MAY 14 1973

March 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

THE LIBRARY OF THE
MAY 2 1973
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

UIUCDCS-R-73-567

CALCULATION OF GFSR PSEUDORANDOM NUMBER BINARY STARTING MATRIX

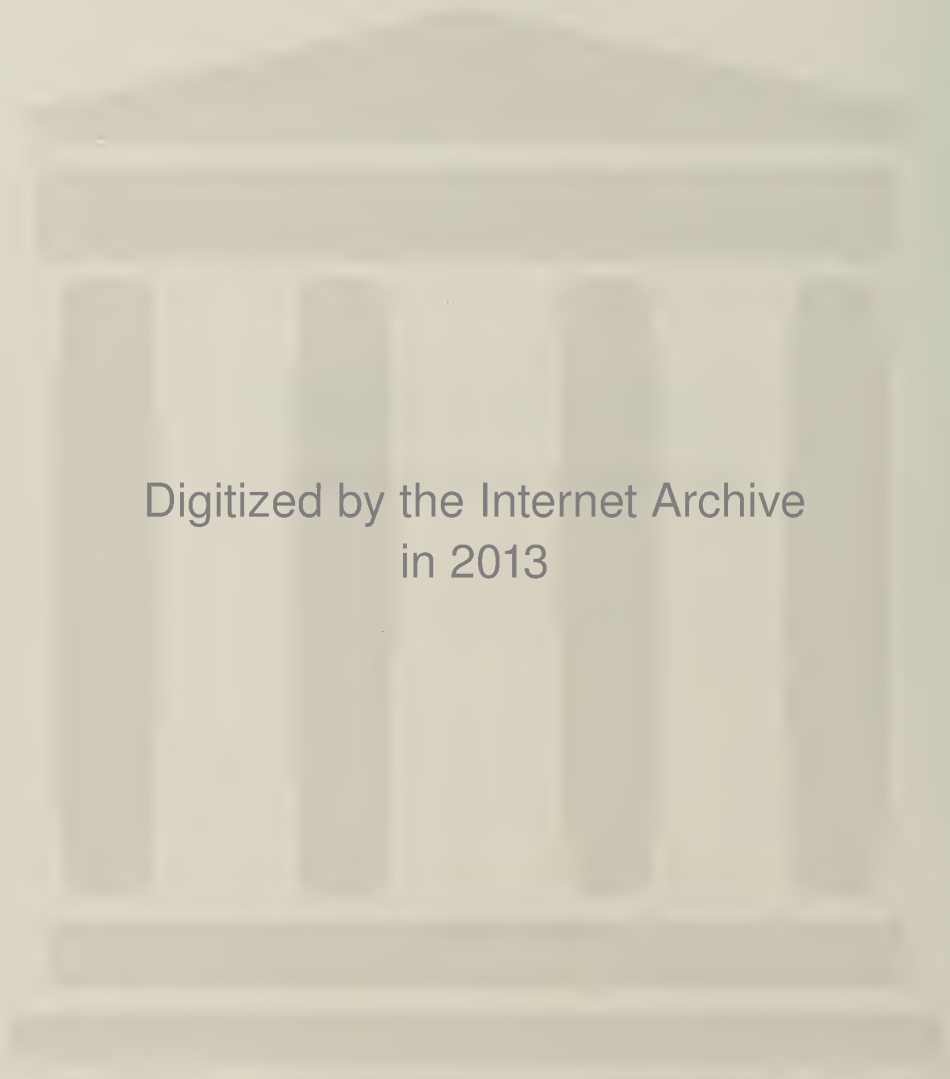
by

W. H. Payne*

March 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

* Visiting Research Associate Professor of Computer Science on leave from the Computer Science Department and the Computing Center, Washington State University, Pullman, Washington 99163.



Digitized by the Internet Archive
in 2013

http://archive.org/details/calculationofgfs567payn

1. INTRODUCTION

The generalized feedback shift register (GFSR) pseudorandom number generator produces the 'same' floating-point sequence of multi-dimensional distributed pseudorandom numbers on any computer. The 'same' sequence in the sense that high-order bits agree on all machines. Not only is the period of the pseudorandom sequence independent of the word size of the host machine, but the algorithm can be implemented with one exclusive-or ($\oplus$) and two add (modulo p) instructions. This makes its execution extremely fast for mini- or microcomputers [1].

The pseudorandom number generator is based on a primitive trinomial modulo 2 of the form $X^P + X^Q + 1$ and a resulting feedback shift register sequence.

Statement of the GFSR algorithm is:

GFSR Pseudorandom Number Algorithm

Initialize integer J=0

Initialize table M(1),M(2),...,M(P)

Step:

1. J ← J+1
2. If J > P, set J ← 1
3. K ← J+Q
4. If K > P, set K ← 1
5. Exclusive-or, M(J) $\oplus$ M(K)
6. STORE, M(J) ← M(J) $\oplus$ M(K)

The procedure of initializing the table, M(1),M(2),...,M(P) of computer words (up to 98 bits), is a nontrivial time-consuming task (about five minutes 360/75 CPU time). Specifically, a 98 x 98 binary matrix with linearly independent (modulo 2) columns is needed. Further, a judiciously selected delay of the basic feedback shift register sequence between columns to achieve excellent statistical performance of the generator must be sought.

It is the purpose of this paper to present an initial M table, both in octal and hexadecimal, for the polynomial $X^{98} + X^{27} + 1$ (period $2^{98}-1$) computed from the initialization algorithm in [1]. For this specific polynomial, L-space distributed numbers to N-bit accuracy where $LN < 98$ will be produced by the GFSR algorithm. For example, it is possible to generate six-distributed random numbers to 15-bit accuracy since $6 \times 15 = 90 < 98$.

2. THE TABLE

Alignment of both the octal and hexadecimal digits on the 98 bits was set so that the high-order octal digit attains a maximum value of three while the corresponding hexadecimal digit attains a maximum value of seven. This was done to insure that the sign digit was zero for those programmers who wish to implement the algorithm in compiler languages.

J = 0 GFSR PSEUDORANDOM NUMBER GENERATOR
P = 98 STARTING MATRIX (98 X 98) BITS).
Q = 27 OCTAL HEXADECIMAL

M(1) =	05121565254D6473332646123D6536210	14A375560D3860698A6357910
M(2) =	10641430472016615060463053353D447	2343189D0381A30998A00624E
M(3) =	3475072507375055115445744522117 23	7301051DFA2026C97C95227A6
M(4) =	21367042137107104146475277713343D	45EE222F91C88669EAFF96E30
M(5) =	32364221674117574756122460733 4D30	69E891DE13EF9EE294C387030
M(6) =	20534361024723523365065247447164	4288F10A74EA6F51AA9E49CE8
M(7) =	02442035031017047655147472465704	0A44100C83C4FAD33CEA4D788
M(8) =	16545064312662322175274570004362	3AC174656C9A48FA8C8CD11E4
M(9) =	27122164012455407112044176547733 5	5CA474054860E4A121FACFD8A
M(10) =	26314016026273747437171124575317	59980E082EFCF1F3C92A5F59E
M(11) =	11542107547002704026152025362774 0	26C44783800881635055E5FC0
M(12) =	30073302017D45144527654073522363	6076C207894C95706DE0049E6
M(13) =	11422261766677552615242767 1D601D1	262481F86FED58D517DC8CD82
M(14) =	45451257667216430244361244444536	5652AF08A3A3UA478A92495CC
M(15) =	17462666020127102561613310175452 4	3E6586081 5C8571C5820F82A8
M(16) =	11645042750424302751372451424303 4	274A22F4451 85E9704A628C38
M(17) =	17327727635742633716275173512370 4	30AF07CEF8837CE5E9EE94F88
M(18) =	04126634766472570507405066331550 7	10AD9CF84EAF147828098368E
M(19) =	04676216551461350535346741152117 5	137C8E84CC501507378404 4FA
M(20) =	0536D0040414714660246525144141172	15E0041DCE66C14055321 84F4
M(21) =	11523405555722561266674174507 5072	26A70586F4AE2860E1F28F474
M(22) =	34352071652136035326516166343240 1	710439051783AD6A7109C6A02
M(23) =	11157211D1U3051311314552002112327	24DE8904314825996A01129AE
M(24) =	14462710406111665U1161154D5000473	3265C8831276A09C4D82B02 76
M(25) =	04265622212707247702216514325434 4	116892457104FC247531A81C8
M(26) =	04354257357510326460545151454776 4	1108AF77C21AD30829A659FE8
M(27) =	37705425131115515 4D7721025450344 0	7F88152C97408U7E885650E4C
M(28) =	16771527223111326460236227141634 24	38F35749920698279 2E61CE28
M(29) =	05640416D710720473 17D722302075123	17410E1C8E84ECF102610UF4A6
M(30) =	21231743217560257251626570403 2565	4533E3470C15EA9C85E206AEA
M(31) =	36240741501630506253240331567416 1	7941E1A0E628CA850366EF0E2
M(32) =	04764602656357U1U047151455310542 5	13E9820738C102734C859162A
M(33) =	10333341106425646131343253166537 4	2186E1234574C5971AACE05F8
M(34) =	01670075272741160942700016734014 5	0770305078 4E162E00388 80CA
M(35) =	37265423360023265D2147606620260 00	706813780406A119F009D5800
M(36) =	35745141110023547663570657057663 6	77CA612404ECF838C68C5FB3C
M(37) =	35630316736235444447411126151361 71	7730CEEF2764927849580 2F72
M(38) =	01172741750024224132550571406664 4	04F5E1F405128 5A845E600848
M(39) =	27554700614440645060005127134121	5E09C0C64834A3002D28970A2
M(40) =	11505743034315120405500130167242 1	2688E3DE334A105A0160EEA22
M(41) =	34655511D7240660642036417457300 34	7358491D418DD1D7A1F2F6D38
M(42) =	13661360041154427753034501232550 6	2F62F0109822FE80E5053568C
M(43) =	12106534155027637366423261306511 2	28806C3685F3EF689AC58D494
M(44) =	32315055305016076665775064213237 2	699A206283870B5FE80116 9F4
M(45) =	23574351234163665004222601051106 1	4EF8E94E1CF6A04496045246 2
M(46) =	33671577551116621501265471630554 6	6F737F8493B23415ACE7316CC
M(47) =	05241601505766671040620270541573 1	154381A2F08722UC82E2C3782
M(48) =	17645637454660453214547272440020 5	3F489F966C2568C83AEA4010A
M(49) =	11123552205513336455427421707 7061	24A76A4202080208C478FC62
M(50) =	03715752732363243262735612754226 5	0F98EAEC3CD4682EEE28D896A
M(51) =	12373000261754510000315466021470 D0	29F6DD58F82900066CD823380
M(52) =	13704125430160714272364043422552 7	2F8855 8C1C398BA734DE256AE
M(53) =	02517670614676775420230022 0D01672	0A9F88C66F8F8104C04800774
M(54) =	30355172764627D2361160556222300 10	610A7AFA65C2789C2DC926010
M(55) =	26742743610405416113737507136342 6	58C5E3C44161C48EF01C8CE2C
M(56) =	23710535644402343724337677257106D	4F915D02409C7046FEFD5E460
M(57) =	16007636512623077734467317306212 1	380F9EA364C7F0C988308C8A2
M(58) =	35127520714D5D0653343005254240471	74AF50E60A06A0C6055628272
M(59) =	33773600711361457026747210255564 1	6FF78DE48C65E16F3A2158742
M(60) =	36235355134536502203575257541 5647	79JAE02E57A84838EABEC374E
M(61) =	31376547652546454012640643230 4407	65F0670559A580A00680312DE
M(62) =	37672773537007246472131166616 0515	7F75F8AF81D4D3A2C9081C29A
M(63) =	11515643675644356544651264310631	2698A30EE9107AC9A95A32332
M(64) =	27137207652564246334403033301 2550	5C8EB7055D14C0C8186082A0D
M(65) =	22020312223517606277332D243466433	4820CA4903F0C8F6D051C0A36
M(66) =	32560763410052371004262241361 1113	6AE1F3840A9F2U459285E2496
M(67) =	33074145224356636D3032545611635 01	6C78654A3883C186AC8B9CE82
M(68) =	13150066760D31663247226746736440 23	2C0036F8338353A58C0D09026
M(69) =	05114703461076333762460245620 6753	1499C3988F987F2B2972180 6
M(70) =	10446233655460550250515573617656	224C9806CC2D0A8A60EF1E75C
M(71) =	30347156723250624257146033621 7310	61CE6EE9AA328AF33066F23D90
M(72) =	24432312513775101330076360441 2353	5234CAA5FF482081F3C242906
M(73) =	30067D5067061523555601545470670 43	606E280C6353B6E06C8380C46
M(74) =	10056401257471450536506 7424616632	205DD157CE6515EA378A63834
M(75) =	23352703721403370675216746507241 2	40D5C3E8C0DF1804779A8EA14
M(76) =	26002374033532231275547411141475 5	58D4FC0DD69328DB3C24C330A
M(77) =	25666705145146344644353275D153057	5760C532999C9A475AF41AC5E
M(78) =	33347250412174240201523613562750 3	60CEA8851F14081A9E2EE5E86
M(79) =	30653476626366455073116760310746	6356E7096784847646EF8323CC
M(80) =	03167505125030061661260250115572 6	DCEF452A866381582A0987AC
M(81) =	12153701205461516165025673644 3540	2807C142CC69C750AEEF48EC0
M(82) =	15342304333225670017125220164510 6	35C4C46DA5770QF2AA40E948C
M(83) =	00367340345543151D654074032146D2 0	U1EEED7208CD2358 3CD01982A
M(84) =	13611D2030430255345404416526515 43	2F121062 30A072C121D56A6CA
M(85) =	12361677051145U0076035245052164 03	29E38F1499401F0754A2A3A06
M(86) =	20070164627642D64535421263576667	4D7U74C8E886950 8A1590F86E
M(87) =	27332231506544132444432522424402 6	508499A359U85248D54A2902C
M(88) =	04624744541214643247064706652550 6	1329E480A3346A71A7185568C
M(89) =	11464366564144206504153135607 2431	2668F68A1910044359770EA32
M(90) =	14677557430507236437074402346 61	337F6F8C51D3U1F1C49327362
M(91) =	24061500027737332552140363374 3210	50634D08F70856A303C0F8010
M(92) =	01203611127417051542647674231 6755	05078928C3C5362D3EF13380A
M(93) =	35111560704357543505716114212 7456	7493706238EC745E713115E5C
M(94) =	16016703134U5074301631341777070 75	3810C32E0A3C60E65C3FF1C7A
M(95) =	16530457377161213172522617503 6634	3A812F7F9C5167AA963E87838
M(96) =	32570565737422413445676206773 3366	6AF175EFC4A17250F218F75EC
M(97) =	31314703340627100415073333521 1063	6599C37065C81001086EA246A
M(98) =	22536462136763170223006024375746 3	4A803227F7CCF0930305 1F8E6A

Table 1

For a PDP-8 (12-bit word size: 11-bit integer) the value M(1) = 0512, M(2) = 1064,...,M(98) = 2253 would be entered. An INTERDATA computer (word size,16 bits: 15-bit integer) requires M(1) = 14A3, M(2) = 2343,...,M(98) = 4ABD. For an IBM 360 (32-bit fullword: 31-bit integer), M(1) = 14A37556, M(2) = 2343189D,..., M(98) = 4ABD322F. For a last example, the CDC 6000 series (word size, 60 bits: integer size, 48 bits; initialized to 47 bits) requires M(1) = 0000051215652540647 3, M(2) = 000010641430472016 61,..., M(98) = 0002253646213676317. A multiple precision routine (of doubtful value) to 98 bits is easily implemented.

3. COMPILER IMPLEMENTATION

Figure 1 presents an example of a non-ANSI standard FORTRAN coding of the GFSR algorithm.

```
      FUNCTION RAND(M,P,Q,INTSIZ)
C M(P) = TABLE OF P PREVIOUS RANDOM NUMBERS.
C P, Q = POLYNOMIAL PARAMETERS: X**P+X**Q+1.
C .NOT. OPERATOR IMPLEMENTED IN ARITHMETIC.
C INTSIZ = INTEGER SIZE (BITS) OF HOST MACHINE: E.G.,
C IBM 360, 31; CDC 6000, 48; SRU 1100, 35; HP 2100, 15.
      LOGICAL AA,BB,LCOMPJ,LCOMPK
      INTEGER A,B,P,Q,INTSIZ,M(1)
      EQUIVALENCE (AA,A),(BB,B),(MCOMPJ,LCOMPJ),(MCOMPK,LCOMPK)
      DATA J/0/
      N=(2**(INTSIZ-1)-1)*2+1
      J=J+1
      IF(J.GT.P) J=1
      K=J+Q
      IF(K.GT.P) K=K-P
      MCOMPJ=N-M(J)
      MCOMPK=N-M(K)
      A=M(K)
      B=M(J)
      BB=LCOMPJ.AND.AA.OR.LCOMPK.AND.BB
      M(J)=B
      RAND=FLOAT(M(J))/FLOAT(N)
      RETURN
      END
```

Figure 1. FORTRAN Implementation of GFSR Pseudorandom Number Generator

This particular coding has been certified for 1) IBM 360, 2) Sperry-Rand 1108, 3) Control Data Corporation 6400, and 4) Hewlett-Packard 2116 computers. The first five GFSR pseudorandom numbers for each of these machines is given in Figure 2. Validity of coding any implementation can be checked against these values.

0.36963295936584470	0.36963297000000000
0.40631365776062010	0.40631372000000000
0.42877840995788570	0.42877845000000000
0.47411382198333740	0.47411389000000000
0.95315784215927120	0.95315778000000000
(a)	(b)

0.36963297409225149	0.36964017152786255
0.40631371808778027	0.40632343292236328
0.42877845193692465	0.42878508567810059
0.47411388879095284	0.47410506010055542
0.95315778681866803	0.95318460464477539
(c)	(d)

Figure 2. The First Five Normalized Pseudorandom Numbers From GFSR: $x^{98} + x^{27} + 1$, delayed column=9800, produced on the a) IBM 360, b) SRU 1108, c) CDC 6400, and d) HP 2116 computers.

4. CONCLUSIONS

The heretofore unachievable feat of essentially duplicating the same probabilistic simulation on any model computer using the fast, general, multidimensional GFSR pseudorandom number generator is now possible.

REFERENCE

[1] Lewis, T. G. and Payne, W. H., "Generalized Feedback Shift Register Pseudorandom Number Algorithm," Journal of the ACM (to appear about August 1973).

APPENDIX A

Method

Let $X^P + X^Q + 1$ be a primitive trinomial modulo 2 such that $Q < P/2$. Binary digits, a_i, obey the recurrence $a_k = a_{k-P+Q} \oplus a_{k-P}$. The companion matrix for polynomial $X^P + X^Q + 1$ is of the form

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & & 0 \\ 0 & 0 & 1 & 0 & \cdots & & 0 \\ \cdot & \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & & \cdot \\ 0 & 0 & \cdot & \cdot & & 1 & 0 \\ 1 & 0 & & & & & \end{bmatrix}$$

All zeros except a 1 in the Q-th column

thus,

$$A \times V_i = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & \cdot & \cdot & \cdot & \\ & & & & & \\ & & & & & \\ & & & & & 0 \\ 1 & & & & & \end{bmatrix} \begin{bmatrix} a_i \\ a_{i+1} \\ \cdot \\ \cdot \\ \cdot \\ a_{i+Q} \\ a_{i+P-1} \end{bmatrix} = \begin{bmatrix} a_{i+1} \\ a_{i+2} \\ \\ \\ \\ \\ a_i \oplus a_{i+Q} \end{bmatrix} = \begin{bmatrix} a_{i+1} \\ a_{i+2} \\ \\ \\ \\ \\ a_{i+P} \end{bmatrix} = V_{i+1}$$

For this initialization application, let $a_0 = a_1 = \ldots = a_{P-1} = 1$. Next form $V_d = A^d \times V_0$. Set $V_0 = U_1$, and with V_d form a N x 2 dimensioned matrix such that V_0 is the first column while V_d is the second column: that is

$$U_2 = V_1 \times [1\ 0] + V_d \times [0\ 1].$$

Form

$$U_3 = U_1 \times [1\ 0\ 0] + A^d \times U_2 \times \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Continue this process. Matrix U_N will be a P x N matrix with each of its columns of the form V_0, V_d, V_{2d},...,$V_{(N-1)d}$. Numerical values in the table presented in this paper were to be both in octal and hexadecimal so a value of N, a multiple of 3 and 4, was selected. Here d = 100 x P = 9800 and N = 24. The first N bits of table M were the rows of matrix U_N.

Let $U_{N,2}$ be the matrix of the second N bits. Clearly these are easily obtained from

$$U_{N,2} = A^{dN} \times U_N$$

For statistical reasons, the generator was cycled another 5000 P times. Order of these operations does not affect the final result so U_N was cycled 5000 P times before calculation of $U_{N,2}$.

```
C GFSR PSEUDORANDOM NUMBER ALGORITHM INITIALIZATION TO 98 BITS.
      INTEGER   M(98,5)
      INTEGER ONE,DELAY,P,Q
      P=98
      Q=27
      N=24
      DELAY=100*P
      ONE=2**(N-1)
      DO 1 I=1,P
1     M(I,1)=ONE
      DO 2 J=1,N
      DO 3 K=1,DELAY
3     X=IXOR(M,P,Q,N)
      DO 4 L=1,P
4     M(L,1)=M(L,1)/2+ONE
2     CONTINUE
      DO 5 I=1,5000
      DO 5 J=1,P
5     X=IXOR(M,P,Q,N)
      DO 6 I=2,5
      DO 10 K=1,P
10    M(K,I)=M(K,I-1)
      DO 7 K=1,N
      DO 7 J=1,DELAY
7     X=IXOR(M(1,I),P,Q,N)
6     CONTINUE
      DO 11 I=1,P
11    M(I,1)=M(I,1)-(M(I,1)/ONE)*ONE
      WRITE(6,8)((M(I,J),J=1,5),I=1,98)
8     FORMAT('0',5Z8)
C FORMAT ROUTINE FOR CONVERSION TO OCTAL AND HEX READS FROM UNIT 9.
      WRITE(9,9)((M(I,J),J=1,5),I=1,98)
9     FORMAT(5Z8)
      STOP
      END
```

```
FUNCTION IXOR(M,P,Q,NN)
LOGICAL AA,BB,LCOMPJ,LCOMPK
INTEGER A,B,P,Q,M(1)
EQUIVALENCE (AA,A),(BB,B),(MCOMPJ,LCOMPJ),(MCOMPK,LCOMPK)
DATA J/0/
N=2**NN-1
J=J+1
IF(J.GT.P)J=1
K=J+Q
IF(K.GT.P) K=K-P
MCOMPJ=N-M(J)
MCOMPK=N-M(K)
A=M(K)
B=M(J)
BB=LCOMPJ.AND.AA.OR.LCOMPK.AND.BB
M(J)=B
IXOR=M(J)
RETURN
END
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-73-567	2.	3. Recipient's Accession No.
4. Title and Subtitle: CALCULATION OF GFSR PSEUDORANDOM NUMBER BINARY STARTING MATRIX			5. Report Date: March 1973 6.
7. Author(s): W. H. Payne			8. Performing Organization Rept No.
9. Performing Organization Name and Address: Department of Computer Science, University of Illinois, Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address			13. Type of Report & Period Covered: Individual Research 14.

15. Supplementary Notes

16. Abstracts

The generalized feedback shift register (GFSR) pseudorandom number algorithm produces the same floating-point sequence of pseudo-random numbers on any computer. The algorithm requires a binary initialization matrix. A matrix, given both in octal and hexadecimal, for primitive polynomial, modulo 2, $x^{98} + x^{27} + 1$ is listed.

17. Key Words and Document Analysis. 17a. Descriptors

Random Numbers
Simulating
Monte Carlo
Shift Register Sequences

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement: unlimited distribution	19. Security Class (This Report): UNCLASSIFIED	21. No. of Pages: 14
	20. Security Class (This Page): UNCLASSIFIED	22. Price

OCT 29 1973

BOUND BY
HW

UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 no.565-570(1973
Two upper bounds for the weighted peth l
3 0112 088400624